$$\begin{tikzcd} & \textbf{Polymorphism} \arrow[dl] \arrow[dr] & \\ \text{Method Overloading} & & \text{Method Overriding} \end{tikzcd}$$

Polymorphism

Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So, polymorphism means many forms.

Polymorphism

It refers to the use of a single type entity (method, operator or object) to represent different operations (types) in different scenarios.

Polymorphism

We can achieve polymorphism in Java using the following ways:

- Method Overloading
- Method Overriding

Polymorphism

We can achieve polymorphism in Java using the following ways:

- Method Overloading :

    This is an example of compile time (static polymorphism or early binding)

- Method Overriding :

    This is an example of runtime time (Dynamic Method Dispatch or late binding)

Polymorphism

Method Overloading

It is used to achieve compile-time polymorphism (Early binding). It allows us to use the same method name but different signatures. If a class has more than one method with the same name but a different method signature, it is known as method overloading.

Polymorphism

Advantage of method overloading

Method overloading increases the readability of the program.

```
static float order(float total){
    return total;
}
```

```
static float order(float total, float deliveryCosts){
    return total + deliveryCosts;
}
```

```
static float order(float total, float deliveryCosts, String promo){
    return total + deliveryCosts - 2;
}
```

Polymorphism uses those methods to perform different tasks. This allows us to perform a **single action in different ways**

Polymorphism

Method Overriding
(Dynamic Binding or Late Binding)

It is used to achieve run-time polymorphism or Dynamic Method Dispatch (late binding).

Polymorphism

Method Overriding
(Dynamic Binding or Late Binding)

Rules for Java Method Overriding:

1. There must be an IS-A relationship (inheritance).
2. The method must have the same name as in the parent class .
3. The method must have the same parameter as in the parent class .

Polymorphism

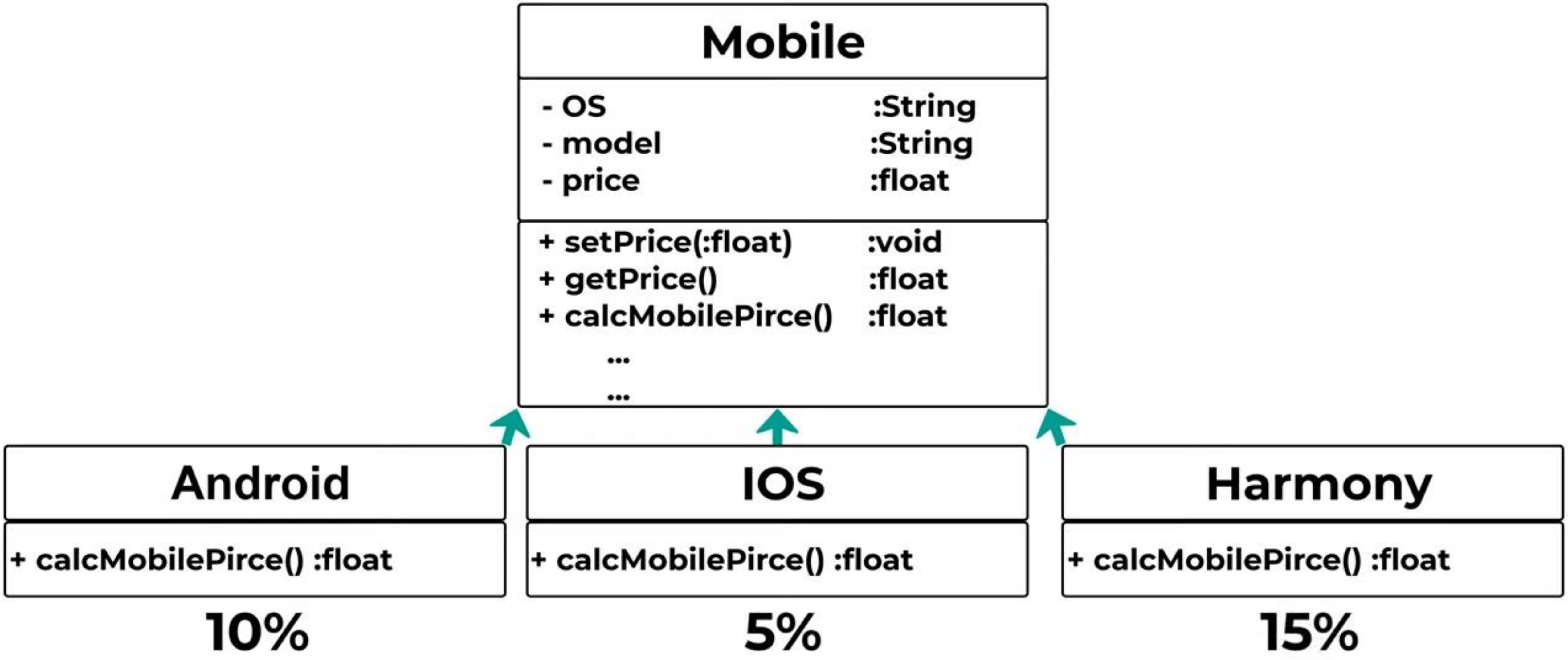
Method Overriding

```
public class SuperClass{
      void method(){
           ...
      }
}

public class SubClass extends SuperClass{
      void method(){
           ...
      }
}
```

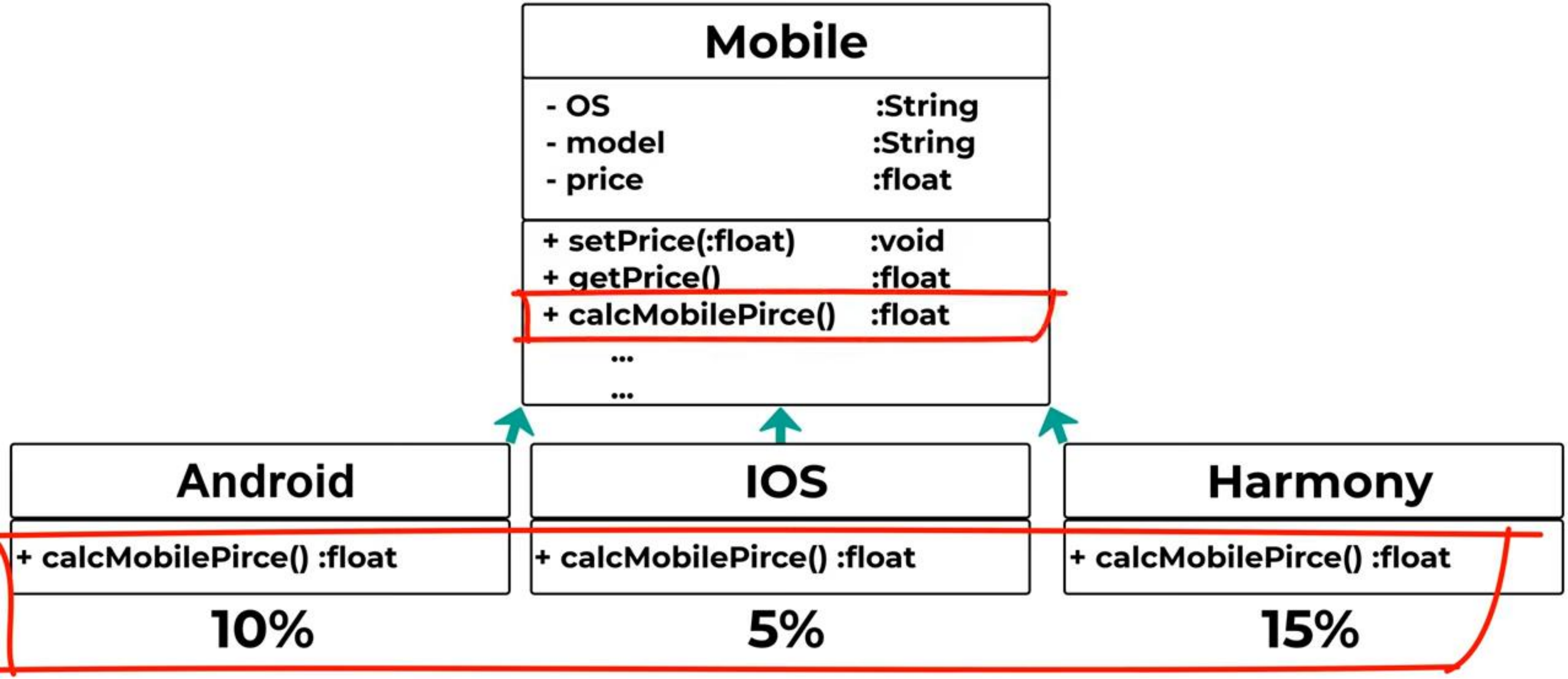
Polymorphism Method Overriding

Mobile Shop Management System



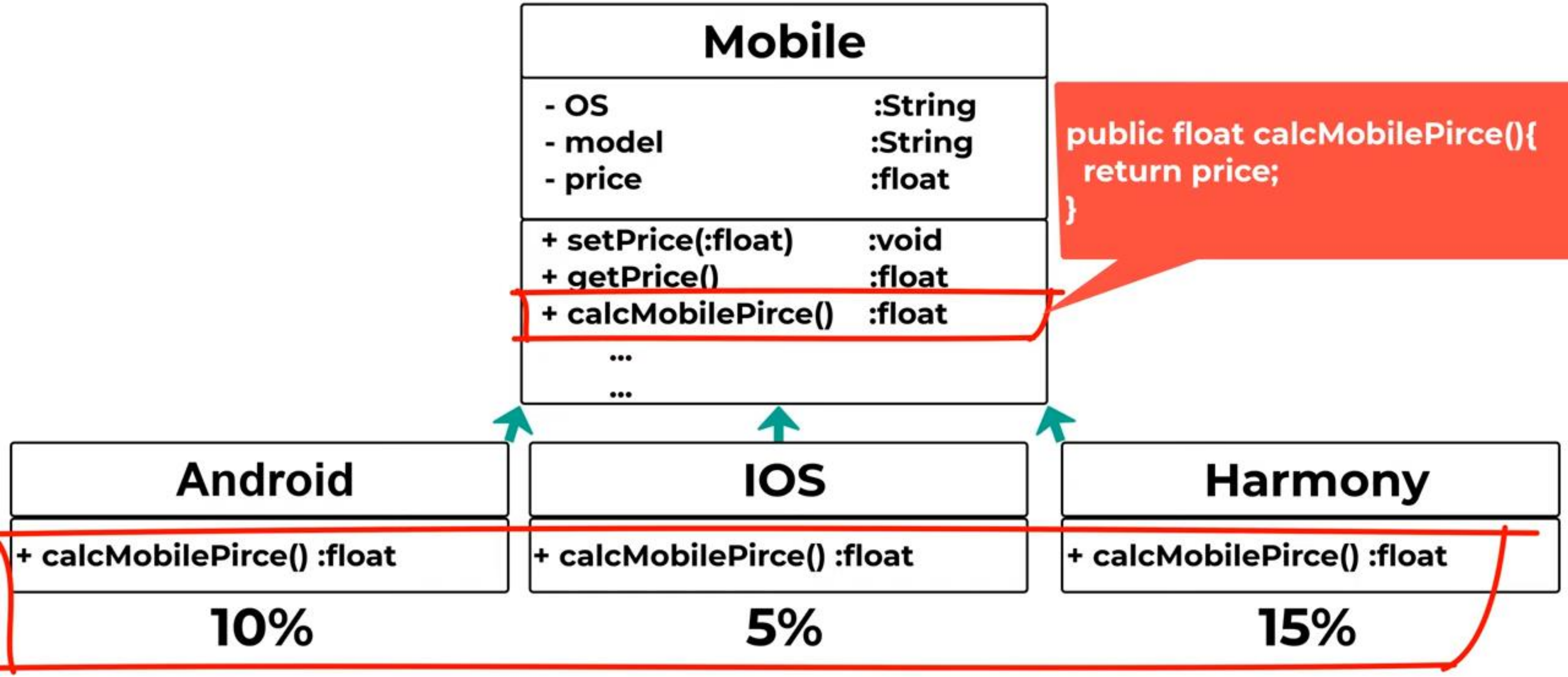
Polymorphism Method Overriding

Mobile Shop Management System



Polymorphism Method Overriding

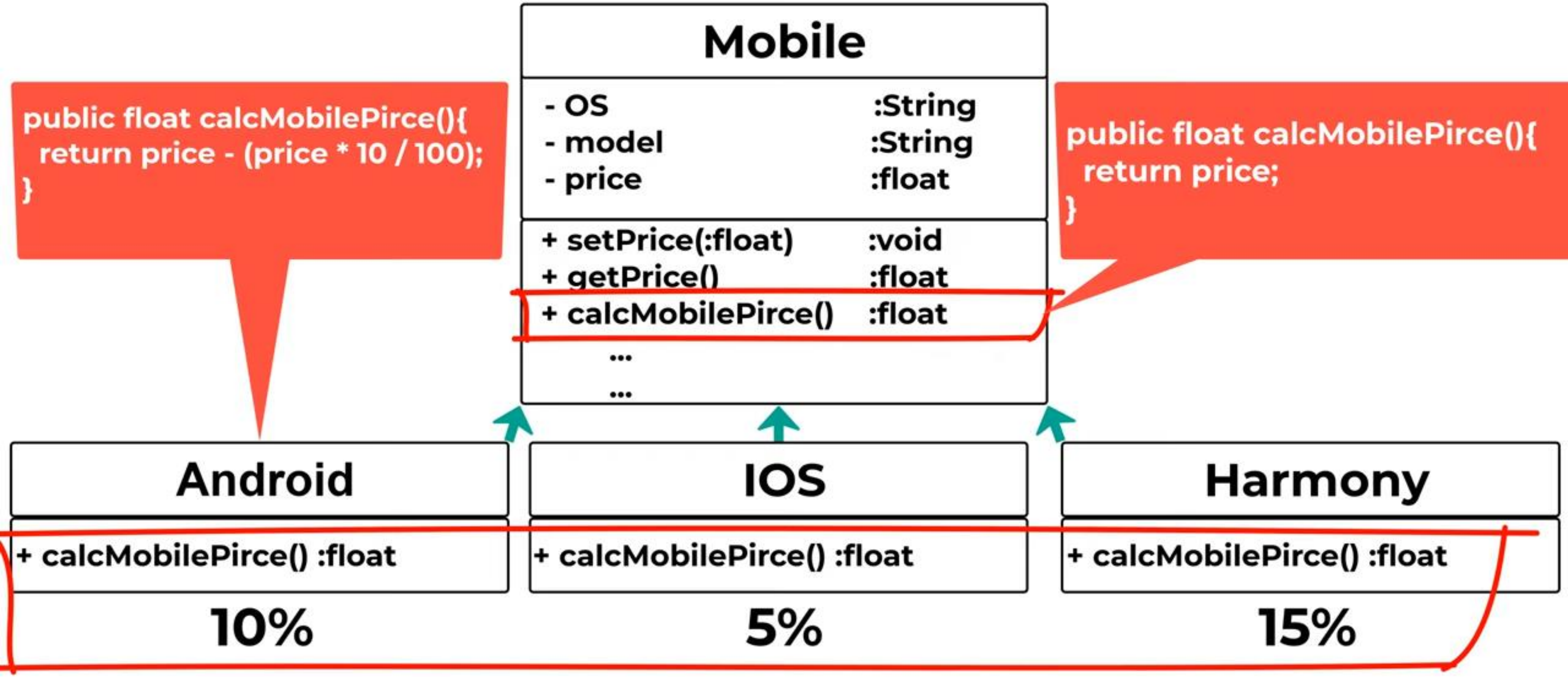
Mobile Shop Management System



```
public float calcMobilePirce(){
  return price;
}
```

Polymorphism Method Overriding

Mobile Shop Management System

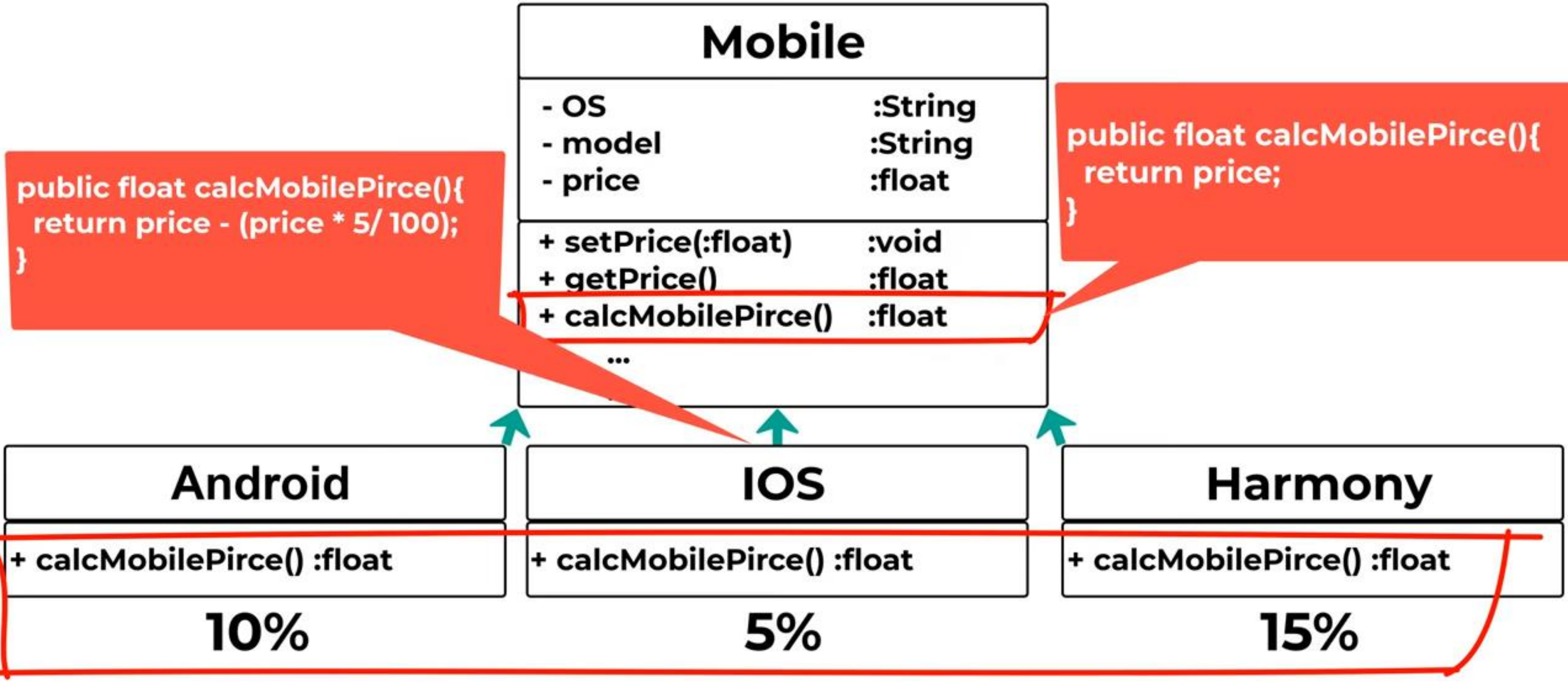


```
public float calcMobilePirce(){
  return price - (price * 10 / 100);
}
```

```
public float calcMobilePirce(){
  return price;
}
```

Polymorphism Method Overriding

Mobile Shop Management System

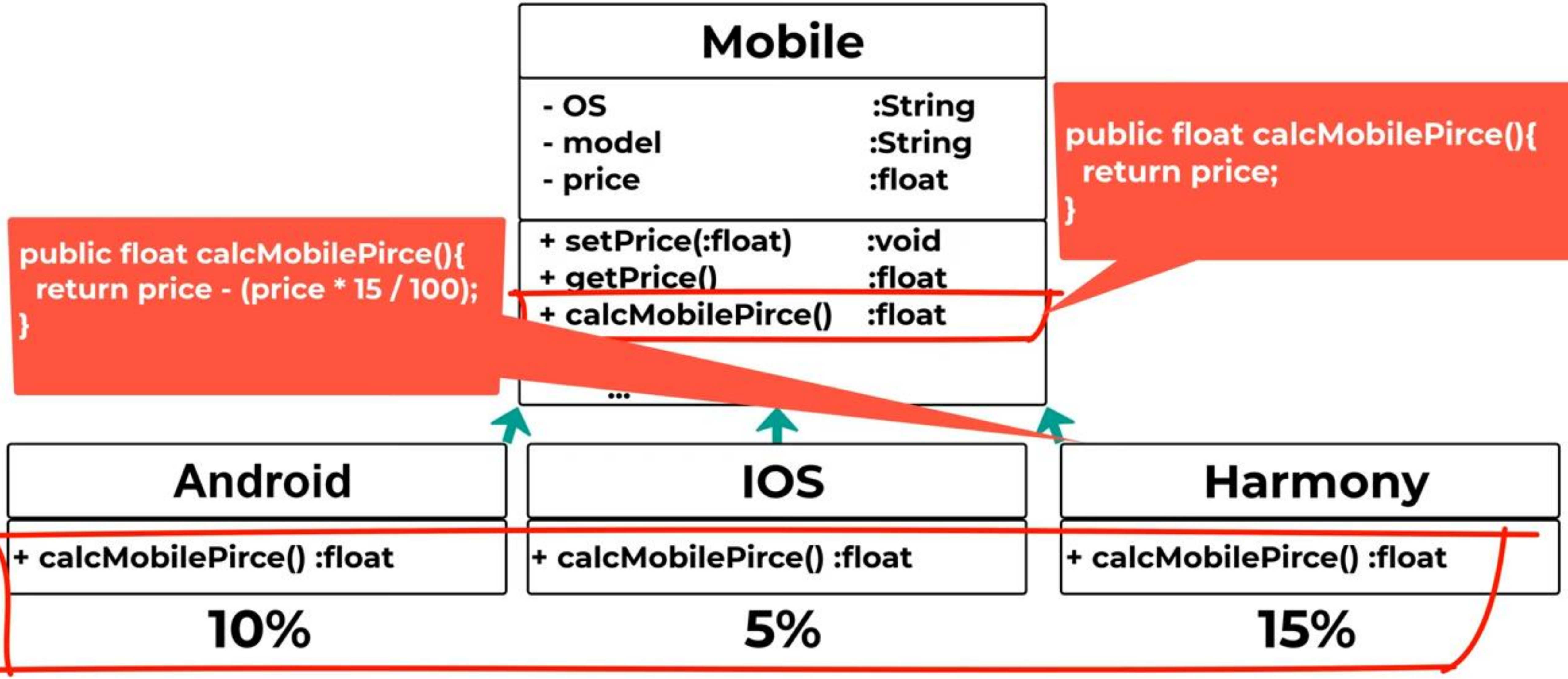


```
public float calcMobilePirce(){
  return price - (price * 5/ 100);
}
```

```
public float calcMobilePirce(){
  return price;
}
```

Polymorphism Method Overriding

Mobile Shop Management System

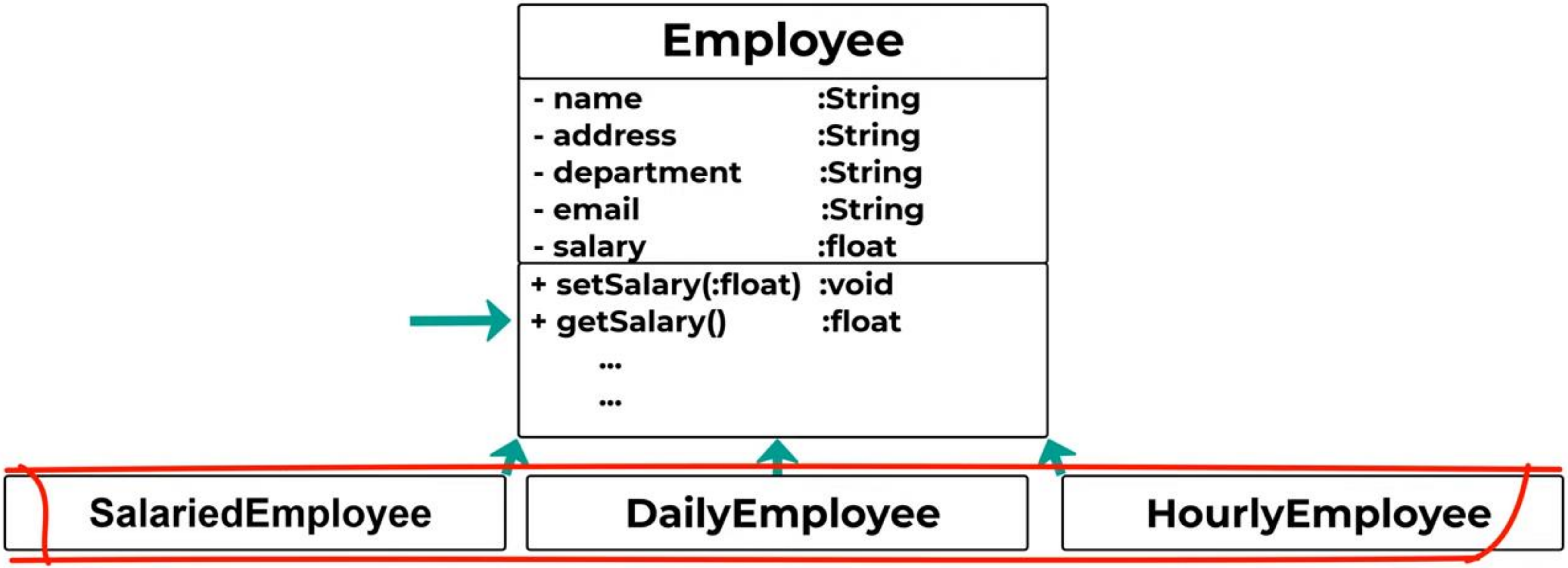


```
public float calcMobilePirce(){
  return price;
}
```

```
public float calcMobilePirce(){
  return price - (price * 15 / 100);
}
```

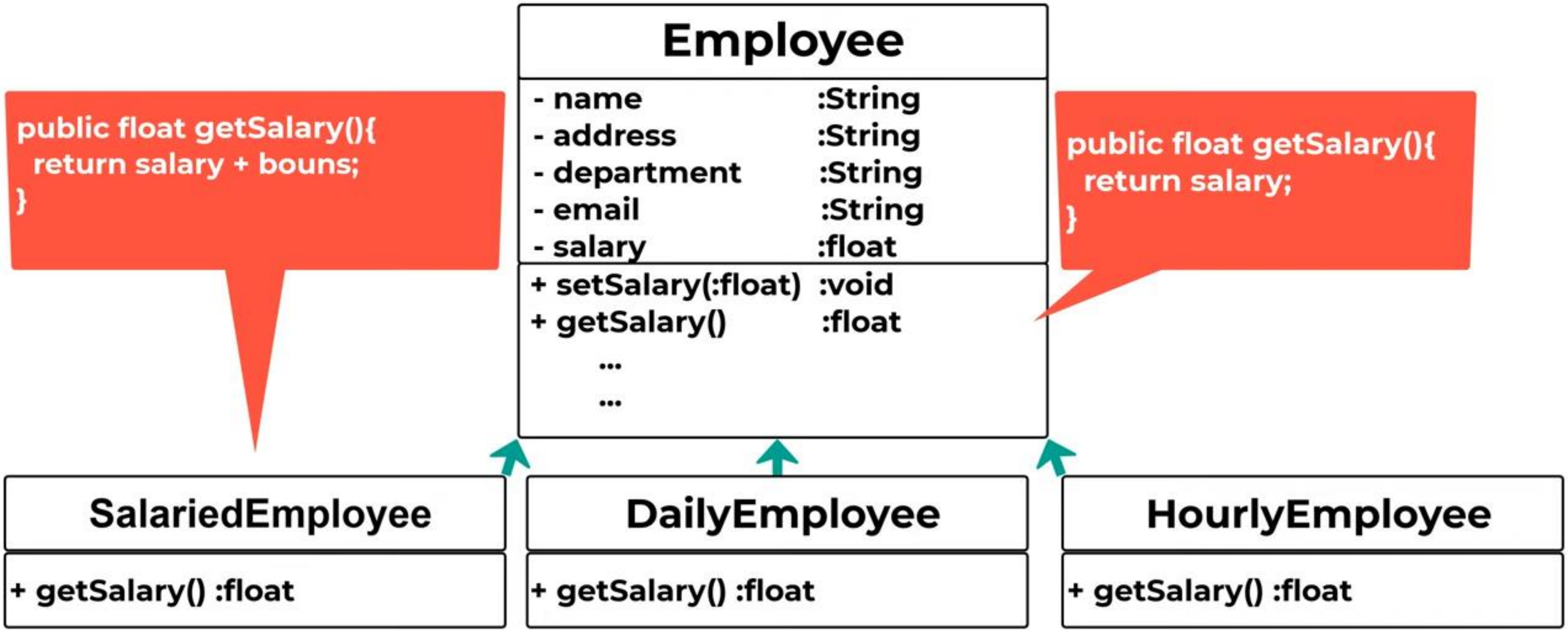
Polymorphism Method Overriding

Employee Payroll Management System



Polymorphism Method Overriding

Employee Payroll Management System

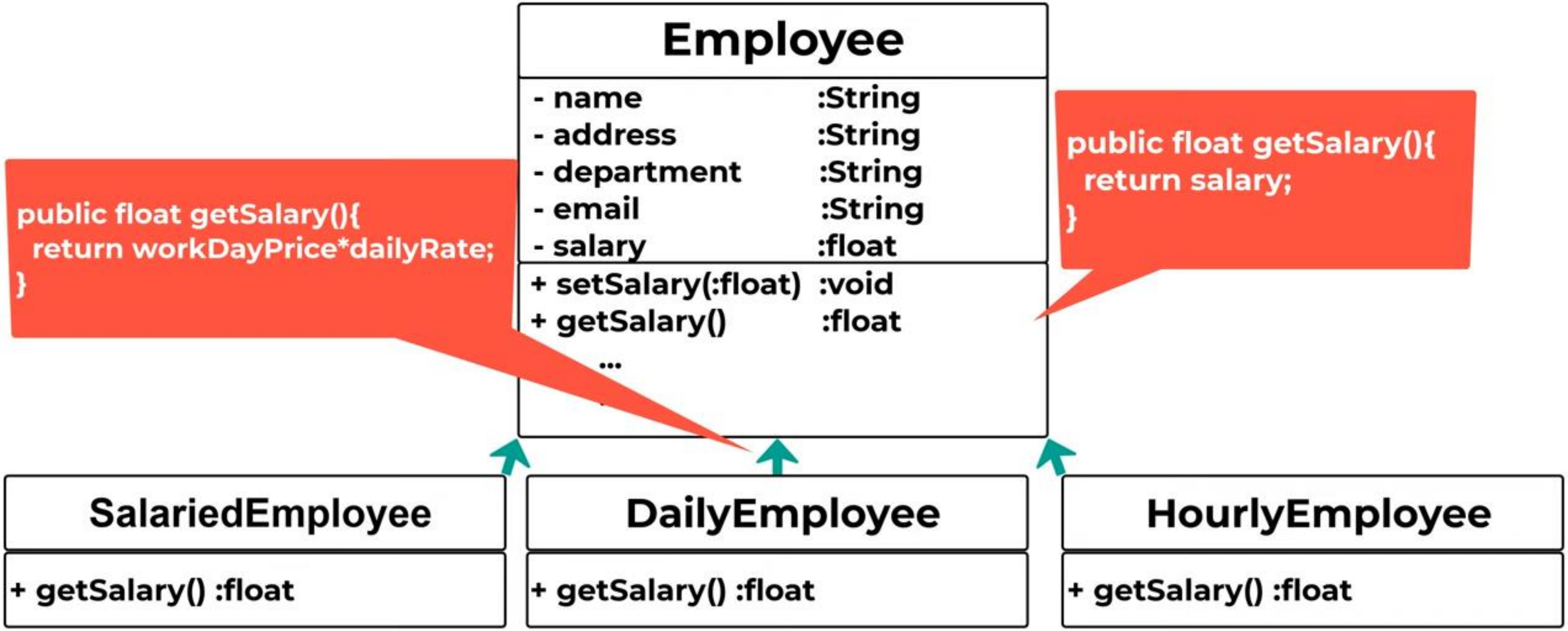


```
public float getSalary(){
  return salary + bouns;
}
```

```
public float getSalary(){
  return salary;
}
```

Polymorphism Method Overriding

Employee Payroll Management System

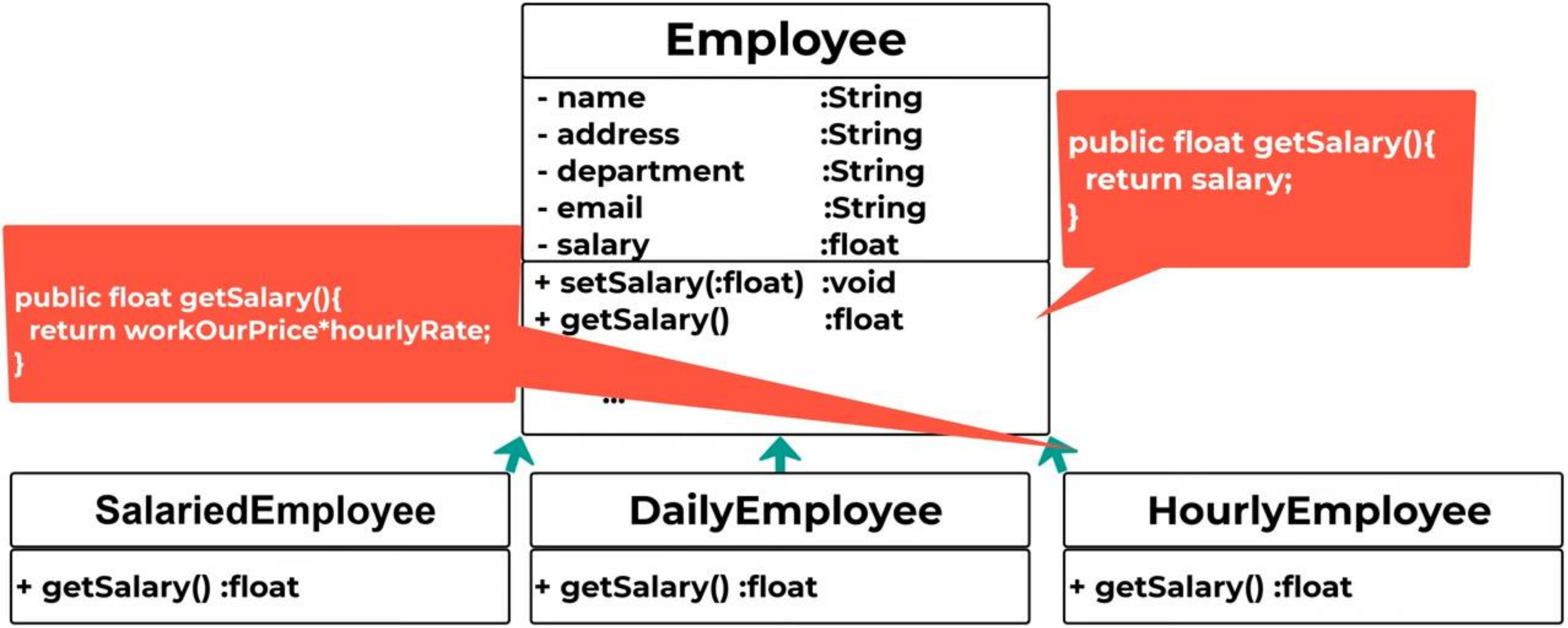


```
public float getSalary(){
  return workDayPrice*dailyRate;
}
```

```
public float getSalary(){
  return salary;
}
```

Polymorphism Method Overriding

Employee Payroll Management System



```
public float getSalary(){
  return salary;
}
```

```
public float getSalary(){
  return workOurPrice*hourlyRate;
}
```